Preliminary Design Specifications

RTCS DQ and RC Control Panel Replacement

March 2, 1993

1.0 Introduction

This manual is intended to provide detailed design specifications for use in the development of a new DQ console control system. It will also form the basis for the future program as-built specifications.

Requirements for the DQ console control have been reviewed and a baseline was approved on September 30, 1992. An implementation has been suggested using Vivisun programmable buttons, 2 monochrome terminals, and a color console. The initial target platform is a 486 PC, however the design is not computer specific. The specifications are written with the assumption that the DQ console control system will reside on a dedicated computer, however it could reside concurrently on another computer such as the data system computer or one of the display systems.

1.1 Directory Structure

The system name will be DQ, and files used in system development will be kept under a directory of that name. Directory DQ will contain source and header files for the main DQ program. A separate subdirectory will be created under directory DQ to contain files for each child program. A program subdirectory will have the same name as the child whose files it contains. A special LIB subdirectory will contain files for functions which are used by more than one of the programs. A library will be created with those functions.

1.2 Multi-tasking

In order to allow for independent asynchronous I/O on several different devices, multiple process will be used. In general, each program will perform only one physical input or output with wait in its main processing loop. The operating system will be given the responsibility of swapping from program to program as events occur. Most programs will communicate to each other through message queues.

1.3 Compilation and Linking

Each program and the library will have a make file in its directory to compile and link it. In addition the DQ directory will have a special make file to compile and link all the programs which

make up the system.

2.0 Procedure Descriptions for DQ Programs

This section contains a description with program design language for each of the DQ programs.

2.1 DQ - DQ Control Console Main Program

The DQ main program will be the driver for the DQ console control system. It will be started up automatically when the computer is booted assuming a dedicated computer system. It will initialize all parameters, tables, queues, and shared memory, and will start all the children programs. It will then wait for a halt signal. When a halt signal is received, it will terminate the children programs. It will then go through the initialization process again and restart the children programs and wait. There will be no keyboard input.

```
DQ start
Initialize parameters, tables, queues, shared memory
Do forever (until computer turned off)
        Start children programs in priority order
        Wait for halt signal
        Kill children programs
Enddo
DQ stop
```

2.2 BOn - Button Ouput Programs

There will be a button output program for every Vivisun controller card. Each controller card will control up to 16 Vivisun programmable buttons. When started by DQ a BOn program will initialize all parameters and tables. It will initialize the appropriate RS-232 serial port and notify the corresponding BIn program, then enter its main processing loop. The main processing loop will receive messages from PR. It will be queue suspended until a message is received. A message will tell BOn which button is to be affected and how. It may tell which one of a predefined set of messages to write out the port, or it may contain information which needs to be encoded into the proper button format before being written out the port. BOn will then loop back to receive the next message. It will only be halted when killed by DQ.

```
BOn start
Initialize parameters and tables
Initialize RS-232 serial port
If successful
        Send port initialized signal to BIn
Else
        Do what ??? try several times ???
```

```
        Send error message to DO
        ??? Go to sleep until killed ???
Endif
Do forever (until killed)
        Read message queue
        If message with data
                Encode data into button format
        Else
                Identify predefined message
        Endif
        Write message out serial port
Enddo
BOn stop
```

2.3 BIn - Button Input Programs

There will be a button input program corresponding to every BOn program. When started by DQ a BIn program will initialize all parameters and tables. It will wait for a signal from its corresponding BOn that the RS-232 serial port has been initialized, then enter its main processing loop. The main processing loop will read an ASCII character from the serial port. It will be I/O suspended until a character is received. If the character indicates that a button has been pushed, BIn will send a message to PR telling which button was pushed, otherwise, BIn will ignore it. BIn will then loop back to read the next character. It will only be halted when killed by DQ.

```
BIn start
Initialize parameters and tables
Wait for port initialized signal from BOn
Do forever (until killed)
        Read ASCII character
        If button pushed
                Send message to PR
        Endif
Enddo
BIn stop
```

2.4 HB - Heartbeat Interval Timer Program

The heartbeat interval timer program will be set to wake up at a predefined time interval. It may signal other programs which need to know that the time has elapsed, or set the time in shared memory. It will be used for such things as to determine if communications have been lost between the computers, or to know when to poll interfaces such as the countdown clocks.

```
HB start
Initialize
Do forever (until killed)
        Sleep
```

```
        Send message to affected programs
        Set time in shared memory
Enddo
HB stop
```

2.5 SI - Serial Input Program

There will be a serial input program to receive messages from the data processing computer system. When started by DQ the SI program will initialize all parameters and tables. It will initialize the appropriate RS-232 serial port and notify SO that it has been initialized, then enter its main processing loop. The main processing loop will read messages sent by the data processing computer system an ASCII character at a time from the serial port. It will be I/O suspended until a character is received. Each time a complete message is identified, it will be sent to PR. SI will only be halted when killed by DQ.

```
SI start
Initialize parameters and tables
Initialize serial port
If successful
        Send port initialized signal to SO
Else
        Do what ??? try several times ???
                (Could be a bad interface, but at any rate the system is effectively dead,
and should be halted.)
Endif
Do forever (until killed)
        Read serial port
        If start of message
                Clear message area
                Set byte counter to 0
                If message in progress
                        Send error message to DO
                Endif
                Set message in progress flag
        Else if end of message
                If message in progress
                        Send message to PR
                        Clear message in progress flag
                Else
                        Send error message to DO
                Endif
        Else (middle of a message)
                If message in progress
                        Increment byte counter
                        Store message byte
                Else
                        Send error message to DO
```

```
            Endif
        Endif
Enddo
SI stop
```

2.6 SO - Serial Output Program

There will be a serial output program to send messages to the data processing computer. The messages will essentially say that a particular button was pushed. When started by DQ the SO program will initialize all parameters and tables. It will wait for a signal from SI that the RS-232 serial port has been initialized, then enter its main processing loop. The main processing loop will receive messages from PR. It will be queue suspended until a message is received. When a message is received, it will be written out the serial port. SO will then loop back to receive the next message. It will only be halted when killed by DQ.

```
SO start
Initialize parameters and tables
Wait for port initialized signal from SI
Do forever (until killed)
        Read message queue
        Write message out serial port.
Enddo
SO stop
```

2.7 EI - Ethernet Input Program

There will be an Ethernet input program to receive messages and data buffers from both the data processing and graphics display computer systems. When started by DQ the EI program will initialize all parameters and tables. It will check the Ethernet hardware and communication protocols and notify EO that connections they are correct. It will establish the Ethernet connection, then enter its main processing loop. The main processing loop will read messages from the graphics display computer system or data buffers from the data processing computer system. It will be Ethernet I/O suspended until a buffer is received. When a buffer is received it will be identified as a graphics message or a data buffer. Graphics messages will be sent to PR. Data buffers will be sent to TO. EI will then loop back to receive the next buffer. It will only be halted when killed by DQ.

```
EI start
Initialize parameters and tables
Check hardware and protocols
If successful
        Send Ethernet OK signal to EO
Else
        Do what ??? try several times ???
        Send error message to DO
        ??? Go to sleep until killed ???
```

```
Endif
Establish Ethernet connection
If unsuccessful
        Do what ??? try several times ???
        Send error message to DO
        ??? Go to sleep until killed ???
Endif
Do forever (until killed)
        Read Ethernet buffer
        If data buffer
                Queue to TOn (may have to use shared memory for speed ??? )
        Else if graphics message
                Send message to PR
        Endif
Enddo
EI stop
```

2.8 EO - Ethernet Output Program

There will be an Ethernet output program to send messages to the graphics display system. The messages will essentially say that a particular button was pushed. When started by DQ the EO program will initialize all parameters and tables. It will wait for a signal from EI that the Ethernet hardware and protocols are correct. It will establish the Ethernet connection, then enter its main processing loop. The main processing loop will receive messages from PR. It will be queue suspended until a message is received. When a message is received, it will be encoded into an Ethernet buffer and written out the Ethernet port. EO will then loop back to receive the next message. It will only be halted when killed by DQ.

```
EO start
Initialize parameters and tables
Wait for Ethernet OK signal from EI
Establish Ethernet connection
If unsuccessful
        Do what ??? try several times ???
        Send error message to DO
        ??? Go to sleep until killed ???
Endif
Do forever (until killed)
        Read message queue
        Encode message into Ethernet buffer
        Write message out Ethernet port
Enddo
EO stop
```

2.9 RI - Relay Input Program

There will be a relay input program to determine the status of the mechanical relays. This description assumes that the relays will not send an interrupt when they change state, but must be polled. When started by DQ the RI program will initialize all parameters and tables. It will do whatever needs to be done to establish communications ??? with the relays, then enter its main processing loop. The main processing loop will receive wake-up signals from HB. When signalled it will poll the relays and check to see if any have changed state. If the state is determined to be changed, the new state will be sent in a message to PR. RI will then loop back and wait until signalled again. It will only be halted when killed by DQ.

```
RI start
Initialize parameters and tables
Initialize relay states to -1
Clear state changed counters
Establish relay communication
If successful
        Send communication established signal to RO
Else
        Do what ??? try several times ???
        Send error message to DO
        ??? Go to sleep until killed ???
Endif
Do forever (until killed)
        Wait for wake-up signal from HB
        Do for all relays
                Read relay state (0 or 1)
                If the state changed from the previous state
                        Increment state changed counter
                        If state changed counter > noise filter
                                Save new state
                                Clear state changed counter
                                Send new state message to PR
                        Endif
                Endif
        Enddo
Enddo
RI stop
```

2.10 RO - Relay Output Program

There will be a relay output program to send fire signals to the mechanical relays. Some type of capability must be built in (in hardware???) to save a fire signal which was sent to an unarmed relay. The capability must also be given to clear a fire signal queued up for an unarmed relay. When started by DQ the RO program will initialize all parameters and tables. It will wait for a signal from RI that communication with the relays has been established, then enter its main processing loop. The main processing loop will receive messages from PR. It will be queue suspended until a message is received. A message will tell RO which relay is to be fired or cleared. It will fire or clear the relay then loop back to receive the next message. It will only be

halted when killed by DQ.

```
RO start
Initialize parameters and tables.
Wait for communications established signal from RI
Do forever (until killed)
        Read message queue
        If fire message
                Issue fire command to relay
                If relay armed
                        Command is transmitted
                Else
                        Hardware saves command and it will automatically be
                                transmitted when armed
                Endif
        Else if clear message
                Clear fire transmit hardware
        Endif
Enddo
RO stop
```

2.11 TI - Time Input Program

There will be a time input program which will use two time channels on a NASA Data Receive/Transmit Card. The card will not send an interrupt but must be polled. When started by DQ the TI program will initialize all parameters and tables. It will initialize the NASA DRT card, then enter its main processing loop. The main processing loop will receive wake-up signals from HB. When signalled it will poll the time channel on the NASA DRT card for each mission being processed. The mission time(s) and GMT will be sent to DO for display. TI will then loop back and wait until signalled again. It will only be halted when killed by DQ.

```
TI start
Initialize parameters and tables
Initialize NASA DRT card
If unsuccessful
        ??? Try several times???
        Send error message to DO
        ??? Go to sleep until killed???
Endif
Do forever (until killed)
        Wait for wake-up signal from HB
        Read channel 1 for CDT and GMT
        If unsuccessful
                Send error message to DO
        Else
                Send GMT time message to DO
                Send CDT1 time message to DO
```

```
        Endif
        If mission 2 defined
                Read channel 2 for CDT
                If unsuccessful
                        Send error message to DO
                Else
                        Send CDT2 time message to DO
                Endif
        Endif
Enddo
TI stop
```

2.12 TOn - Terminal Output Programs

There will be a terminal output program to display Ethernet data values on each of the two monochrome terminals. When started by DQ a TO program will initialize its terminal port then enter its main processing loop. The main processing loop will receive data buffers from EI. It will be queue suspended until a data buffer is received. When a data buffer is received, the data values will be decoded (??? here or in EI or in a program that does no I/O ???) and displayed on the monitor. TOn will then loop back to receive the next data buffer. It will only be halted when killed by DQ. (How to do screen writing to avoid flicker? Possible a screen writer like curses.)

```
TOn start
Initialize parameters and tables
Initialize terminal port
If unsuccessful
        ???Try several times???
        Send error message to DO
        Set terminal out flag
Endif
Do forever (until killed)
        Read data queue (or shared memory queue ???)
        Decode data
        Check for errors and send error messages to DO ???
        Format output for terminal
        Send data to proper terminal
Enddo
TOn stop
```

2.13 DO - Display Output Program

There will be a display output program to display titles, times, indicators, and messages on the color display console. Information to display may come from any of the other programs. When started by DQ the DO program will initialize all parameters and tables. It will establish communication with the color display console, then enter its main processing loop. The main processing loop will receive information in a message queue from other programs. It will be

queue suspended until a message is received. A message will tell which part of the display to change and how. It may tell which one of a set of predefined messages to display, or it may contain information which needs to be encoded into the proper format before being displayed. DO will then loop back to receive the next message. It will only be halted when killed by DQ.

DO start
Initialize parameters and tables
Establish communications with monitor
If unsuccessful
    ??? Try several times???
    This is DO, so it doesn't help to send it a message to display
    Should we log messages then???
    Do we want to be able to stop error display or logging??? YES, added a new
    button to enable and disable logging ???
Endif
Do forever (until killed)
    Read message queue
    Determine message type
    If invalid message
        Display error message
    Endif
    Format output for display
    Display output on monitor
Enddo
DO stop

2.14 PR - Processor Program

There will be a processor program which forms the core of the control console system. The processor program will basically be a message processor. It will receive messages from the other programs, determine which programs or computer systems need to know about the messages, and pass them on. It will also keep track of the state of the system and be responsible for changes in the system state. Each button will have a response lag time associated with it. Pushing a button will not cause it to light, but will cause a message to be sent to the processor. The processor will notify the appropriate system and wait for a response before telling a BO program to light the button. If the button is pushed again before the response arrives, the second push will be ignored, unless the response lag time has elapsed. (How do we guarantee that the console matches the actual status of the other systems in case of message loss? Should this be a set of programs on the data system???)

The processor will work as an engine for a finite state machine. Each button will have a discrete number of possible ways it can be lit, and a known number of events that can affect it. For each lit state and each event, the button can only transition to one other lit state. Identifying all the possible states is crucial to the processor design. After it processes a message, PR will loop back to receive the next message. It will only be halted when killed by DQ.

PR start

```
Initialize parameters and tables
Clear response lag time table
Do forever (until killed)
        Read message queue
        If button pushed message
                If response lag time elapsed
                        Send message to data and/or graphics system or PR queue
                                ???may need to process some itself???
                        Set response lag time
                Endif
        Else if heartbeat message
                Decrement response lag time table entries
                Clear data received indicators
        Else if event message
                Do for all buttons affected by event
                        Send light message to BO
                        Clear response lag time
                        Modify light state table
                        Modify system state table when applicable
                Enddo
        Else if other valid message
                Modify system state table
                Send any information to DO
        Else
                Send error message to DO
        Endif
Enddo
PR stop
```

2.15 EL - Error Logging Program (Do you want one??? If so, it should be able to be inactivated, so a button has been added.)

3.0 Messages and Data

This section lists the possible messages and data to and from the data and graphics systems, along with the section of the Functional Requirements Specification where each is referenced.

3.1 Possible Event Messages to the Processor

Button X pushed

Raw data recording enabled in premission for mission 1	4.1.1
Raw data recording enabled in premission for mission 2	4.1.1
Raw data recording started for mission 1	4.1.1
Raw data recording started for mission 2	4.1.1

Raw data recording stopped for mission 1	4.1.1	
Raw data recording stopped for mission 2	4.1.1	
Processed data recording enabled in premission for mission 1	4.1.2	
Processed data recording enabled in premission for mission 2	4.1.2	
Processed data recording started for mission 1	4.1.2	
Processed data recording started for mission 2	4.1.2	
Processed data recording stopped for mission 1	4.1.2	
Processed data recording stopped for mission 2	4.1.2	
Mission 1 loaded		4.1.3
Mission 2 loaded		4.1.3
Mission 1 started processing data	4.1.3	
Mission 2 started processing data	4.1.3	
Mission 1 stopped processing data	4.1.3	
Mission 2 stopped processing data	4.1.3	
Data system ready	4.2.1	
All missions stopped		
Graphics system ready	4.2.2	
Display X available		
Data system active	4.2.5	
Graphics system active		4.2.6
Relay control manual, software, or both	4.3.2	
Fired relay		4.3.2
Clear fire from unarmed relay	4.3.2	
Relay state indicator	4.3.2	
Data system initialization of mission 1 started	4.4.1	
Data system initialization of mission 2 started	4.4.1	
Data system initialization of mission 1 complete	4.4.1	
Data system initialization of mission 2 complete	4.4.1	
Data source X available		4.5.1
Best for mission 1 available	4.5.1	
Best for mission 2 available	4.5.1	
Next best for mission 1 available	4.5.1	
Next best for mission 2 available	4.5.1	
Add source X, best or next best to display	4.5.1	
Remove source X, best or next best from display	4.5.1	
Alphnumerics for display X frozen	4.5.2.1	
Alphanumerics for display X activated	4.5.2.1	

Manual scaling up for display X activated	4.5.2.2, 4.5.2.3
Manual scaling down for display X activated	4.5.2.2, 4.5.2.3
Manual scaling for display X disabled	4.5.2.2, 4.5.2.3
Data source X available to the RTSP	4.6.1
Data source X not available to the RTSP	4.6.1
Heartbeat time interval elapsed	4.6.2
Data source X selected for processing	4.6.2 - 4.6.5
Data source X not selected for processing	4.6.2 - 4.6.5
Data source X included in best select algorithm	4.6.3
Data source X not included in best select algorithm	4.6.3
Filter initialization for data source X startedi	4.6.4
Filter initialization for data source X completed	4.6.4
Filter for data source X changed to coast	4.6.5
Filter for data source X changed to powered flight	4.6.5
Filter initialization for all mission 1 data sources started	4.6.6
Filter initialization for all mission 2 data sources started	4.6.6
Filter initialization for all mission 1 data sources completed	4.6.6
Filter initialization for all mission 2 data sources completed	4.6.6
Powered flight change for all mission 1 coast filters started	4.6.7
Powered flight change for all mission 2 coast filters started	4.6.7
Powered flight change for all mission 1 coast filters completed	4.6.7
Powered flight change for all mission 2 coast filters completed	4.6.7
Panel test started	4.7.1
Panel test stopped	4.7.1
Buttons at intensity X	4.7.3
Graphics at intensity X	4.7.4
Panel terminals at intensity X	4.7.5
Message logging enabled	4.7.6
Message logging disabled	4.7.6
Sync error occurred on data source X	4.8.1
Sync error count cleared on data source X	4.8.1
CRC error occurred on data source X	4.8.2
CRC error count cleared on data source X	4.8.2

Sync lock obtained for data source X	4.8.3	
Sync lost for data source X	4.8.3	

3.2 Possible Data Messages to the Processor

Live mission title		4.2.3
Real mission title		4.2.3
Released software version number, title	4.2.4	
Test load modules title		4.2.4
GMT		4.2.7
Countdown time for mission 1		4.2.8
Countdown time for mission 2		4.2.8
Liftoff time for mission 1	4.2.9	
Liftoff time for mission 2	4.2.9	
Software definable function designator	4.4.2	
Data source name		4.5.1
Viewport 1 scale number for display group X	4.5.2.4	
Viewport 2 scale number for display group X	4.5.2.5	
Ethernet data buffer for data source X	4.9	
Error message text		

3.3 Possible Outputs from Processor to Data System

Start recording raw data for mission 1	4.1.1	
Start recording raw data for mission 2	4.1.1	
Stop recording raw data for mission 1	4.1.1	
Stop recording raw data for mission 2	4.1.1	
Start recording processed data for mission 1	4.1.2	
Start recording processed data for mission 2	4.1.2	
Stop recording processed data for mission 1	4.1.2	
Stop recording processed data for mission 2	4.1.2	
Start processing data for mission 1	4.1.3	
Start processing data for mission 2	4.1.3	
Stop processing data for mission 1	4.1.3	
Stop processing data for mission 2	4.1.3	
Liftoff detected for mission 1	4.2.9	
Liftoff detected for mission 2	4.2.9	
Reinitialize mission 1	4.4.1	
Reinitialize mission 2	4.4.1	
Software definable function X pushed		4.4.2

Make data source X unavailable		4.6.1		
Make data source X available		4.6.1		
Stop processing data source X			4.6.2	
Start processing data source X			4.6.2	
Remove data source X from the best select algorithm		4.6.3		
Add data source X to the best select algorithm		4.6.3		
Initialize filter for data source X		4.6.4		
Change filter for data source X to coast		4.6.5		
Change filter for data source X to powered flight		4.6.5		
Initialize filter for all mission 1 data sources	4.6.6			
Initialize filter for all mission 2 data sources	4.6.6			
Change filter for all mission 1 data sources to powered flight		4.6.7		
Change filter for all mission 2 data sources to powered flight		4.6.7		

3.4 Possible Outputs from Processor to Graphics System

Add data source X, best or next best to display group		4.5.1	
Remove data source X, best or next best to display group	4.5.1		
Freeze alphanumerics on display group		4.5.2.1	
Unfreeze alphanumerics on display group		4.5.2.1	
Enable manual scaling on display group		4.5.2.2, 4.5.2.3	
Disable manual scaling on display group		4.5.2.2, 4.5.2.3	
Scale VP 1 of display group up to next predefined scale		4.5.2.2, 4.5.2.4	
Scale VP 1 of display group down to next predefined scale	4.5.2.3, 4.5.2.4		
Scale VP 2 of display group up to next predefined scale		4.5.2.2, 4.5.2.5	
Scale VP 2 of display group down to next predefined scale	4.5.2.2, 4.5.2.5		
Increase intensity of graphics display		4.7.4	
Reduce intensity of graphics displays		4.7.4	

4.0 Detailed Button Message Processing

This section gives the state table for each of the buttons processed by PR finite state machine.

4.1 RECORD RAW (2)

Light States:

0 = off = nothing
1 = white = not recording raw data N
2 = green = recording raw data N

Inputs:

a = raw data recording enabled in premission for mission X
b = raw data recording started for mission N
c = raw data recording stopped for mission N

Transitions:

0 ----- a ----- 1 0 ----- b ----- x 0 ----- c ----- x
1 ----- a ----- x 1 ----- b ----- 2 1 ----- c ----- x
2 ----- a ----- x 2 ----- b ----- x 2 ----- c ----- 1

4.2 RECORD PROC (2)

Light States:

0 = off = nothing
1 = white = not recording processed data N
2 = green = recording processed data N

Inputs:

a = processed data recording enabled in premission for mission N
b = processed data recording started for mission N
c = processed data recording stopped for mission N

Transitions:

0 ----- a ----- 1 0 ----- b ----- x 0 ----- c ----- x
1 ----- a ----- x 1 ----- b ----- 2 1 ----- c ----- x
2 ----- a ----- x 2 ----- b ----- x 2 ----- c ----- 1

4.3 MISSION START (2)

Light States:

0 = off = nothing
1 = white = mission X loaded
2 = green = data processing for mission X in progress
3 = red = data processing for mission X stopped

Inputs:

a = mission X loaded
b = mission X started processing data

c = mission X stopped processing data

Transitions:

0 ----- a ----- 10 ----- b ----- x0 ----- c ----- x
1 ----- a ----- x1 ----- b ----- 21 ----- c ----- x
2 ----- a ----- x2 ----- b ----- x2 ----- c ----- 3
3 ----- a ----- 13 ----- b ----- x3 ----- c ----- x

4.4 FIRE (8)

Light States:

0 = off = nothing / relay state
1 = control type / relay state
2 = fired / relay state

Inputs:

a = relay control manual, software, or both
note: save if new control type comes in while fired
b = fired relay note: save control type to restore if fire cleared
c = clear fire from unarmed relay
note: will only happen if circuit was unarmed
d = relay state indicator
note: display only, doesn't really affect light state

Transitions:

0 ----- a ----- 10 ----- b ----- 20 ----- c ----- x
1 ----- a ----- 11 ----- b ----- 21 ----- c ----- x
2 ----- a ----- 22 ----- b ----- x2 ----- c ----- 1

??? Does the data system actually "fire" the relays or does the control system ??? Probably the control system? What kind of hardware can be used to make this work???

4.5 TIME INIT

Light States:

0 = off = nothing
1 = white = mission X initialized
2 = green = initialization of mission X in progress

Inputs:

a = data system ready
b = data system initialization of mission X started
c = data system initialization of mission X completed

Transitions:

0 ----- a ----- 10 ----- b ----- x0 ----- c ----- x

1 ----- a ----- x 1 ----- b ----- 2 1 ----- c ----- x
2 ----- a ----- x 2 ----- b ----- x 2 ----- c ----- 1

4.6 Data Select for SRC (12), Display Group (6)

Light States:
0 = off = nothing
1 = white data source X name = not selected for display group M
2 = green data source X name (ie. R3) = selected for display group M,

Inputs:
a = data source X (name) available
b = add data source X to display group M (increment number of sources)
b' = add data source X to display group M (at maximum number of sources)
c = remove data source X from display group M

Transitions:
0 ----- a ----- 1 0 ----- b ----- x 0 ----- b' ----- x 0 ----- c ----- x
1 ----- a ----- x 1 ----- b ----- 2 1 ----- b' ----- x 1 ----- c ----- x
2 ----- a ----- x 2 ----- b ----- x 2 ----- b' ----- x 2 ----- c ----- 1

4.7 BEST Select (2) for Display Group (6)

Light States:
0 = off = nothing
1 = white best for mission X name = not selected for display group M
2 = green best for mission X name = selected for display group M

Inputs:
a = best for mission X (name) available
b = add best for mission X to display group M (increment number of sources)
b' = add best for mission X to display group M (at maximum number of sources)
c = remove best for mission X from display group M

Transitions:
0 ----- a ----- 1 0 ----- b ----- x 0 ----- b' ----- x 0 ----- c ----- x
1 ----- a ----- x 1 ----- b ----- 2 1 ----- b' ----- x 1 ----- c ----- x
2 ----- a ----- x 2 ----- b ----- x 2 ----- b' ----- x 2 ----- c ----- 1

4.8 NBST Select (2) for Display Group (6)

Light States:
0 = off = nothing

1 = white next best for mission X name = not selected for display group M
2 = green next best for mission X name = selected for display group M

Inputs:
a = next best for mission X (name) available
b = add next best for mission X to display group M (increment number of sources)
b' = add next best for mission X to display group M (at maximum number of sources)
c = remove next best for mission X from display group M

Transitions:
0 ----- a ----- 1 0 ----- b ----- x 0 ----- b' ----- x 0 ----- c ----- x
1 ----- a ----- x 1 ----- b ----- 2 1 ----- b' ----- x 1 ----- c ----- x
2 ----- a ----- x 2 ----- b ----- x 2 ----- b' ----- x 2 ----- c ----- 1

4.9 ALPHA FREEZE (4)

Light States:
0 = off = display not available
1 = green = display active
2 = white = display frozen

Inputs:
a = display available
b = alphanumerics for display X frozen
c = alphanumerics for display X activated

Transitions:
0 ----- a ----- 1 0 ----- b ----- x 0 ----- c ----- x
1 ----- a ----- x 1 ----- b ----- 2 1 ----- c ----- x
2 ----- a ----- x 2 ----- b ----- x 2 ----- c ----- 1

4.10 VP(2) SCALE for Display Group (4)

Light States:
0 = off = nothing
1 = scale number

Inputs:
a = viewport X scale number for display group M

Transitions:
0 ----- a ----- 1
1 ----- a ----- 1

4.11 MANUAL UP for Display Group (4)

Light States:
0 = off = not manually scaling up
1 = green = manually scaling up

Inputs:
a = manual scaling up for display group X activated
b = manual scaling for display group X disabled
c = manual scaling down for display group X activated

Transitions:
0 ----- a ----- 1 0 ----- b ----- x 0 ----- c ----- x
1 ----- a ----- x 1 ----- b ----- 0 1 ----- c ----- 0

4.12 MANUAL DOWN for Display Group (4)

Light States:
0 = off = not manually scaling down
1 = green = manually scaling down

Inputs:
a = manual scaling down for display group X activated
b = manual scaling for display group X disabled
c = manual scaling up for display group X activated

Transitions:
0 ----- a ----- 1 0 ----- b ----- x 0 ----- c ----- x
1 ----- a ----- x 1 ----- b ----- 0 1 ----- c ----- 0

4.13 DATA CAP (12)

Light States:
0 = off = nothing
1 = green = data source X available
2 = white = data source X not available

Inputs:
a = data source X available to the RTSP
b = data source X not available to the RTSP

Transitions:
0 ----- a ----- 1 0 ----- b ----- x
1 ----- a ----- x 1 ----- b ----- x
2 ----- a ----- 1 2 ----- b ----- x

4.14 DATA PROC (12)

Light States:

0 = off = nothing
1 = green = data source X selected for processing, waiting for data
2 = green = received data source X
3 = blinking green = data source X dropped out
4 = white = data source X not selected for processing

Inputs:

a = data source X selected for processing
b = data source X not selected for processing
c = Ethernet data buffer for source X (Put mode on screen, not on button.)
d = heartbeat time interval reached

Transitions:

0 ----- a ----- 1 0 ----- b ----- 4 0 ----- c ----- x 0 ----- d ----- x
1 ----- a ----- x 1 ----- b ----- 4 1 ----- c ----- 2 1 ----- d ----- 3
2 ---- a ----- x 2 ----- b ----- 4 2 ----- c ----- x 2 ----- d ----- 1
3 ----- a ----- x 3 ----- b ----- 4 3 ----- c ----- 2 3 ----- d ----- x
4 ----- a ----- 1 4 ----- b ----- x 4 ----- c ----- x 4 ----- d ----- x

4.15 BEST SEL (12)

Light States:

0 = off = nothing
1 = red = data source X not in best select algorithm
2 = green = data source X in best select algorithm

Inputs:

a = data source X included in best select algorithm
b = data source X not included in best select algorithm
c = data source X selected for processing
d = data source X not selected for processing

Transitions:

0 ----- a ----- x 0 ----- b ----- x 0 ----- c ----- 1 0 ----- d ----- x
1 ----- a ----- 2 1 ----- b ----- x 1 ----- c ----- x 1 ----- d ----- 0
2 ----- a ----- 1 2 ----- b ----- 1 2 ----- c ----- x 2 ----- d ----- 0

Note: If data was in best select, then processing stopped, then restarted, the source will come up not in best select. The data system will

4.16 FIL INIT (12)

Light States:

0 = off = nothing / data source X not being processed
1 = white = data source X being processed
2 = green = data source X filter initialization in progress

Inputs:

a = data source X selected for processing
b = data source X not selected for processing
c = filter initialization for data source X started
d = filter initialization for data source X completed

Transitions:

0 ----- a ----- 1 0 ----- b ----- x 0 ----- c ----- x 0 ----- d ----- x
1 ---- a ----- x 1 ----- b ----- 0 1 ----- c ----- 2 1 ----- d ----- x
2 ----- a ----- 0 2 ----- b ----- x 2 ----- c ----- x 2 ----- d ----- 1

4.17 PWRD FLT (12)

Light States:

0 = off = nothing
1 = green = data source X being processed with powered flight filter
2 = white = data source X being processed with coast flight filter

Inputs:

a = filter for data source X changed to powered flight
b = filter for data source X changed to coast
c = data source X selected for processing
d = data source X not selected for processing

Transitions:

0 ----- a ----- x 0 ----- b ----- x 0 ----- c ----- 1 0 ----- d ----- x
1 ---- a ----- x 1 ----- b ----- 2 1 ----- c ----- x 1 ----- d ----- 0
2 ----- a ----- 1 2 ----- b ----- x 2 ----- c ----- x 2 ----- d ----- 0

4.18 MA FIL INIT (2)

Light States:

0 = off = nothing
1 = white = filter initializations complete for mission X
2 = green = filter initializations started for mission X

Inputs:

a = filter initialization for all mission X data sources completed
b = filter initialization for all mission X data sources started

Transitions:

0 ----- a ----- 1 1 ----- b ----- 2
1 ----- a ----- x 1 ----- b ----- 2
2 ----- a ----- 1 2 ----- b ----- x

4.19 MA PWRD FLT (2)

Light States:

0 = off = nothing
1 = white = filter switches to powered flight complete for mission X
2 = green = filter switches to powered flight started for mission X

Inputs:

a = powered flight change for all mission X coast filters completed
b = powered flight change for all mission X coast filters started

Transitions:

0 ----- a ----- 1 1 ----- b ----- 2
1 ----- a ----- x 1 ----- b ----- 2
2 ----- a ----- 1 2 ----- b ----- x

4.20 PANEL TEST

Light States:

0 = off = panel test not active
1 = zero = panel test started
2 = panel test message number = panel test in progress

Inputs:

a = panel test started
b = panel test number
c = panel test stopped

Transitions:

0 ----- a ----- 1 0 ----- b ----- x 0 ----- c ----- x
1 ----- a ----- x 1 ----- b ----- 2 1 ----- c ----- 0
2 ----- a ----- x 2 ----- b ----- x 2 ----- c ----- 0

Note: To function in real-time, the panel test processing will increment the panel test message number and generate a message on the input queue. Therefore other messages could come in and be processed. Panel test would not affect the actual light states, and they would be restored following the panel test. Pressing any button would end the panel test.

4.21 DIMMER

Light States:

0 = DIM = intensity will be adjusted down
1 = BRIGHT = intensity will be adjusted up

Inputs:
a = dimmer button pushed

Transitions:
0 ----- a ----- 1
1 ----- a ----- 2

4.22 PANEL, GRAPHS, or BUTTONS

Light States:
0 = off = nothing
1 = intensity number

Inputs:
a = panel, graphs, or buttons at intensity X

Transitions:
0 ----- a ----- 1
1 ----- a ----- 1

4.23 MSG LOG

Light States:
0 = message logging active
1 = message logging not active

Inputs:
a = message logging disabled
b = message logging enabled

Transitions:
0 ----- a ----- 1 0 ----- b ----- x
1 ----- a ----- x 1 ----- b ----- 1

4.24 SYNC ERROR (12)

Light States:
0 = zero = no sync errors
1 = error count number
2 = maximum error count

Inputs:
a = sync error occurred on data source X (increment count)

a' = sync error occurred on data source X (maximum count reached)
b = sync error count cleared on data source X

Transitions:

0 ----- a ----- 1 0 ----- a' ----- 2 0 ----- b ----- x
1 ----- a ----- 1 1 ----- a' ----- 2 1 ----- b ----- 0
2 ----- a ----- x 2 ----- a' ----- x 2 ----- b ----- 0

4.25 DATA ERROR (12)

Light States:

0 = zero = no data errors
1 = error count number
2 = maximum error count

Inputs:

a = CRC error occurred on data source X (increment count)
a' = CRC error occurred on data source X (maximum count reached)
b = CRC error count cleared on data source X

Transitions:

0 ----- a ----- 1 0 ----- a' ----- 2 0 ----- b ----- x
1 ----- a ----- 1 1 ----- a' ----- 2 1 ----- b ----- 0
2 ----- a ----- x 2 ----- a' ----- x 2 ----- b ----- 0

4.26 LOCK (12)

Light States:

0 = off = data for source X not sync locked
1 = green = data for source X sync locked

Inputs:

a = sync lock obtained for data source X
b = sync lost for data source X

Transitions:

0 ----- a ----- 1 0 ----- b ----- x
1 ----- a ----- x 1 ----- b ----- 0

4.27 SOFTWARE DEFINABLE FUNCTIONS

Display only of software defined text on buttons when active and when pushed.

5.0 Detailed Indicator Message Processing

This section gives the state table for each of the indicator lights and display fields processed by PR finite state machine.

5.1 RTSP READY

Light States:
0 = off = nothing
1 = green = data system ready

Inputs:
a = data system ready
b = all missions stopped

Transitions:
0 ----- a ----- 1 0 ----- b ----- x
1 ----- a ----- x 1 ----- b ----- 0

5.2 GRAPHICS READY

Light States:
0 = off = nothing
1 = green = graphics system ready

Inputs:
a = graphics system ready

Transitions:
0 ----- a ----- 1
1 ----- a ----- x ??? does it ever get unready ???

5.3 SYSYEM MODE (2)

Light States:
0 = off = nothing
1 = green mission X title = live mission
2 = red mission X title = simulated mission

Inputs:
a = mission X title, live mode
b = mission X title, simulation mode

Transitions:
0 ----- a ----- 1 0 ----- b ----- 2

1 ----- a ----- x 1 ----- b ----- 2
2 ----- a ----- 1 2 ----- b ----- x

Note: A new mission can be loaded without halting the system.

5.4 SOFTWARE TEST

Light States:
0 = off = nothing
1 = green release version number = officially released software
2 = red test title = test load modules

Inputs:
a = release version number, title
b = test load module title

Transitions:
0 ----- a ----- 1 0 ----- b ----- 2
1 ----- a ----- x 1 ----- b ----- x
2 ----- a ----- x 2 ----- b ----- x

Note: Load modules cannot be changed without halting the system.

5.5 NO PANEL INPUT

Light States:
0 = off = nothing
1 = green = data system active, waiting for pulse signal
2 = green = data system active, received pulse signal
3 = blinking green = data system dropped out

Inputs:
a = data system active
b = heartbeat time interval elapsed

Transitions:
0 ----- a ----- 2 0 ----- b ----- x
1 ----- a ----- 2 1 ----- b ----- 3
2 ----- a ----- x 2 ----- b ----- 1
3 ----- a ----- 2 3 ----- b ----- x

5.6 NO GRAPHICS INPUT

Light States:
0 = off = nothing

1 = green = graphics system active, waiting for pulse signal
2 = green = graphics system active, received pulse signal
3 = blinking green = graphics system dropped out

Inputs:
a = graphics system active
b = heartbeat time interval elapsed

Transitions:
0 ----- a ----- 2 0 ----- b ----- x
1 ----- a ----- 2 1 ----- b ----- 3
2 ----- a ----- x 2 ----- b ----- 1
3 ----- a ----- 2 3 ----- b ----- x

5.7 GMT

Display only of GMT.

5.8 COUNTDOWN (2)

Display only of countdown time / mission elapsed time for each mission.

5.9 LIFTOFF (2)

Display only of liftoff time for each mission.

5.10 MESSAGES

Display only of error messages or data messages. There will probably be room for a rolling display of multiple lines. It would be desirable to increment a count of a message if it already appears on the screen.

5.11 PLASMA DISPLAYS

Data displays only. Mode will be displayed here instead of on the button itself. Care will be taken to only write changed data in order to avoid flashing of the screen.